Amendments to the Specification

Please replace the paragraph beginning on page 6, line 5, with the following rewritten paragraph:

When an instruction block is fetched from memory, it may contain a combination of single-word and double-word instructions. In this case, the boundaries between the instructions are unknown, so the instruction block must be pre-decoded before the instructions can be entered into the pipeline. The process by which a block of instructions is fetched from memory and prepared for execution by a pipelined microprocessor is illustrated in Fig. ~~1~~2.

Please replace the paragraph beginning on page 7, line 6, with the following rewritten paragraph:

Fig. 3 represents an 8-word instruction block (illustrated beginning at instructions 22c, 22b, and 22a then the forward decoding instructions 20a, 20b, 20c, 20d, and 20e) along a cache line beginning at internal memory address 4000h, and extending to address 4007h. In this case, the microprocessor has requested the instruction at address 4003h, resulting in the entire instruction block being fetched from memory. As stated earlier, before the instructions in this block can be transferred to the cache, the block must be pre-decoded. The block contains a combination of single-word and double-word instructions, and the boundaries of these instructions must be known before the instructions can enter the pipeline.

Please replace the paragraph beginning on page 7, line 14, with the following rewritten paragraph:

The locations within the instruction block are relative to the address (4003h) of the requested instruction I1 20a, which is the value contained in the program counter (PC) of the microprocessor. Conventional pre-decoding proceeds in the forward direction (i.e., toward higher memory addresses) from requested instruction I1 20a, and establishes the addresses of the instructions subsequent to I1. This is easy to do, since there is no ambiguity concerning the size of instructions in the forward direction. For example, beginning with instruction I1 20a, the address of the next instruction I2 20b ~~is~~ can be easily found. The microprocessor can determine from the op code of I1 whether it is a single-word or a double-word instruction. This tells the microprocessor whether instruction I2 20b is located one, or two memory locations away (i.e., PC+1, or PC+2, respectively). In the example of Fig. 3, I1 is a single-word instruction. Therefore, instruction I2 20b begins at address 4004h. Similarly, the op code of I2 indicates

that it is also a single-word instruction, so instruction I3 20c must begin at address 4005h. The op code of instruction I3 reveals it to be a double-word instruction. Therefore, the microprocessor looks for instruction I4 20e two locations away, at address 4007h. Note that this technique works only because we are assured of beginning with the first word of an instruction (namely, the requested instruction, I1). Lacking this information, it would not be possible to resolve the ambiguities of Table 1.

Please replace the paragraph beginning on page 14, line 15, with the following rewritten paragraph:

Fig. 6 compares the structure of single-word and double-word instructions, showing the relationship between the marker and the op code in a double-word instruction, and illustrating the possible ambiguity of the second word of a double-word instruction. A double word instruction is shown, consisting of a first 16-bit word 40a at address n and second 16-bit word 40b at address n+1. For illustrative purposes, the op code of the double-word instruction is assumed to comprise the first 8 bits 42 of the first word 40a. The remainder of the instruction is the operand, which may consist of a memory address. Within the op code 42 of the first word 40a of the double-word instruction is a 4-bit marker sequence 44. This marker sequence, (i.e., 1111) in this particular position within the op code, serves to distinguish the first word of a double-word instruction from all single-word instructions. For example, any single-word instruction 46 with an 8-bit op code 48 has a different bit pattern 50 (i.e., 1100) in the same corresponding position. As noted above, this characteristic makes it possible to distinguish the first word of a double-word instruction from a single-word instruction. A second double-word instruction 52a and 52b is shown in Fig. ~~5~~6. Note that the op code 54 for this instruction differs from that of the pervious double-word instruction 40a and 40b (i.e., 10111101 vs. 10111100). The op code of the first instruction might, for example, denote a "Move from Register to Memory Location" instruction, while the second op code represents a "Jump to Program Location" instruction. Thus, each different double-word instruction has a distinct op code; however, the op code for every double-word instruction must contain the marker bit pattern 44. Note that the second word 52b of the second double-word instruction also (coincidentally) contains the bit marker pattern 44. In this case, the bit pattern 1111 is not part of the op code of the instruction, but simply occurs by chance in the operand (for example, as part of a memory address 0011110000001100). Without prior knowledge that the 16-bit value 52b containing this pattern is the second word (and not the first) of a double-word instruction, the appearance of the marker pattern is ambiguous, and cannot be used to detect the instruction boundary.

Please replace the paragraph beginning on page 15, line 25, with the following rewritten paragraph:

Fig. 7 represents a binary decision tree for an embodiment of the present system and method, in which the size of a cache line is eight 16-bit words. The instruction block 68 in this example contains eight words fetched from the instruction memory at addresses 70 ranging from 4000h-4007h. Recall that an instruction block is fetched in response to a request for an instruction. In this case, the requested instruction (hatched pattern) came from memory location 4004h. The relative location of all eight words in the instruction block with respect to the requested word 74 are expressed in terms of i. For example, the relative location within the instruction block of the requested instruction itself is i, while that of the next word is i+1, and that of the preceding word is i-1 etc. Forward decoding of the words in the instruction block begins with the requested instruction and continues in the direction of higher addresses. Similarly, backward decoding proceeds in the direction of lower addresses. A binary value of 0 or ~~172~~ 1 (designated as numeral 72) is associated with each of the words in the instruction block, depending on whether the respective word contains the double-word marker discussed above.